Please replace the paragraph beginning at page 9, line number 15, with the following rewritten paragraph:

A3

Turning now to Fig. 3, a process 140 for accessing files stored on one of the vaults 130-132 is shown. The process 140 first generates a key from a metadata file (step 142). The metadata file identifies all elements that make up a single application, as identified using a state probe. The operation of the state probe is described in more detail in U.S. Patent No. 5,996,073, entitled "System and Method for Determining Computer Application State," issued on November 30, 1999, the content of which is incorporated by reference.

Please replace the paragraph beginning at page 9, line number 30, with the following rewritten paragraph:

A4

If the component files of the application software to be recreated using the key are large and therefore cumbersome to transfer, it is more efficient to determine whether the component files of the application software already exist locally. Such determination may be made by first looking up the key on the client 122 (step 144) and then optionally comparing the key generated from the metadata file to the key on the client 122 (step 146) and is described in reference to Fig. 4. The key comparison process sets a flag if a difference exists and otherwise clears the flags.

Please replace the paragraph beginning at page 11, line number 8, with the following rewritten paragraph:

A5

The process of Fig. 4 checks whether the sequence attributes are equal (step 166). If so, the difference flag is set (step 168). From step 166, if the attributes are not equal, the process checks whether one of the attributes is newer than the other (step 170). If so, the process proceeds to step 168 to set the difference flag. Alternatively, in the event that the attribute is not newer, the process then checks whether or not one of the attributes is older (step 172). If no, the process proceeds to step 168 to set the difference flag. Alternatively, in the event that one of the attributes is older, the process determines whether or not the file may be overwritten (step 174). If so, the difference flag is set (step 168). Alternatively, the process exits (step 176).

Please replace the paragraph beginning at page 11, line number 21, with the following rewritten paragraph:

A6 Cont

Referring now to Fig. 5, a flowchart illustrating a post-process 180 for placing files onto the vault is shown. The post-process 180 first generates a post key from the metadata file (step 182). Optionally, the process 180 may look up the key present on the vault (step 184) and

A6 Cmcld

compare the keys (step 186). If the comparison causes the difference flag to be set (step 188), the key from the metadata is used to post the file to the vault from the client (step 190). From step 188 or step 190, the process of Fig. 5 exits (step 191). Pseudo-code for the process 180 is as follows:

```
Transform metadata descriptor into key
for each vault
        directly access key on vault
        if found then end for
next
if not found then
        locate first writable vault
        insert file with key
end if
return
```

Please replace the paragraph beginning at page 12, line number 7, with the following rewritten paragraph:

A7

Turning now to Fig. 6, the process 182 of Fig. 5 to generate the post key from the metadata file is shown in more detail. In Fig. 6, metadata associated with each file is generated (step 183). Next, the process 182 verifies the integrity of the file (step 185).

Please replace the paragraph beginning at page 12, line number 21, with the following rewritten paragraph:

A8

A key is then generated from the metadata (step 187) before the process 182 exits (step 189). Key generation should include an integrity checksum as described above as well as basic information about the size, name, and attributes of the file. In the form of a checksum, the key allows identity information as well as integrity information to be easily verified.

Please replace the paragraph beginning at page 13, line number 19, with the following rewritten paragraph:

A9 Cont

Figs. 8A and 8B show alternate processes to provide state-based software life cycle management using a vault. Turning now to Fig. 8A, a process for performing software management first generates the metadata (step 403), as described in Fig. 1. The information is then used to maintain software (step 405) before the process exits. Correspondingly, Fig. 8B shows a second software life cycle management process. Initially, the metadata information is

A9 Cncld.

generated and published (step 410). Next, components of the software are replicated (step 450) based on the metadata. The software is then installed (step 470). After installation, the software may be maintained (step 490).

Please replace the paragraph beginning at page 13, line number 33, with the following rewritten paragraph:

A10

Turning now to Fig. 9, the metadata publication step 410 is shown in more detail. In Fig. 9, a vault is located (step 412). The vault may be a server that maintains items referenced in the metadata files. Next, component files associated with the software are stored in the vault (step 414). Similarly, metadata is stored in the vault (step 416). A catalog, or an index of metadata files that represent the physical components of the software being published, is updated (step 418). Finally, the process 410 exits (step 420).

Please replace the paragraph beginning at page 14, line number 30, with the following rewritten paragraph:

A11

The replicate step 450 of Fig. 8B is shown in more detail in Fig. 11. First, the source vault is located (step 452). Next, the destination vault is located (step 454). Files are then transferred from the source vault to the destination vault (step 456). Similarly, metadata information is copied from the source vault to the destination vault (step 458). Finally, the vault catalog is updated (step 460) before the process of Fig. 11 exits (step 462).

Please replace the paragraph beginning at page 15, line number 3, with the following rewritten paragraph:

A12 Cont

Turning now to Fig. 12, the installation step 470 (Fig. 8B) is shown in more detail in Fig. 12. First, the vault catalog is loaded (step 472). Next, the highest version of the software stored in the vault is determined (step 474). The metadata associated with the highest version of the software is copied to the target machine (step 476). Further, data is remapped (step 478). The process of Fig. 12 then applies a preprocessing operation to the remapped data (step 480) to convert data into the proper format and set up variables appropriately, among others. Further, items associated with the software are installed (step 482). A post-processing process is applied (step 484). This step is similar to step 480 in that variables are checked and data is formatted. Finally, an inventory of the software being installed is updated (step 486) before the process exits (step 488).
